动力学轨迹分析工具（Analysis）简要说明

# 实现依赖

实现语言：C++17　（由于采用了新语法和重度依赖模板机制，暂时只有新版GCC才能正确编译）

构建系统：CMake 3.13

并行实现方式： Intel Thread Building Blocks(TBB) (由于限定于节点内并行，所以未选择MPI，同时程序内部数据结构依赖C++ STL，所以不适合使用OpenMP)

单元测试框架：前期代码使用了Boost.Test, 后期采用Google Test（后者功能更好一些），由于是测试框架是程序一定规模后才使用的，所以目前只是针对后期加入的复杂并且容易出错的代码模块（如语法解析器，中间语言（IR）解释引擎）进行自动测试，同时结合使用gcov对正确性要求高的功能进行代码覆盖率的测试。

通用依赖库：

* Boost (C++准标准库，使用其容器和算法，并且重度使用了其元模板编程库MPL[主要用于自动代码生成以压缩重复冗余的代码片段]和使用Spirit实现语法分析和生成器)
* NetCDF（用于读写Amber的二进制轨迹文件）
* GROMACS dynamic library （用于读写Gromacs的二进制轨迹和拓扑文件）
* fftw3 （傅立叶变换）

# 使用说明

整个程序采用类似Multiwfn的界面交互的设计风格，即主要是通过对程序打印的菜单选择相应的数字以选择相应的功能，命令行参数暂时只是选择程序读写的文件

## 命令行参数

Allowed options:

-h [ --help ] show this help message

-p [ --topology ] topology-file-name topology file

-f [ --file ] trajectory-file-name trajectory file

-o [ --output ] output-file-name output file

--prm tinker-prm-file-name force field file

-x [ --target ] trajectout-file-name target trajectory file

--script script-content script command for non-interactive use

--script-file script-file-name read command from script file

选项说明：

-p 选择拓扑文件，目前支持gromacs的tpr文件，amber的mol2文件和tinker的xyz文件，作为拓扑信息的输入。读取的内容包括原子的名字，类型，残基编号之类的用于标识原子的内容，以及键连关系（用于重建分子结构）

-f 选择轨迹坐标文件，目前支持gromacs的trr和xtc文件，amber的nc文件和tinker的arc文件。若是读取arc文件本身，可以省略-p 输入拓扑文件。 -f 可以指定多次或是连续指定多个轨迹文件，程序将按照命令行指定的顺序连续读取轨迹文件。

例如 :

**analysis -p water.xyz -f water-prod1.nc water-prod2.nc water-prod3.nc**

连续读取三个轨迹文件，效果上等同于先连接三个文件再进行分析

-o 指定分析结果输出的文件。有些功能不需要输入文件，可以不用指定。若是选择的分析功能需要输出，则会在分析模块启动前提示输入文件

--prm 选择力场文件，由于tinker的xyz文件中缺少原子质量电荷等信息，所以需要读取相应的力场文件才能进行相应的分析（例如计算偶极矩需要电荷数据），

若是选择的分析功能需要力场参数文件的内容而未能在命令行参数中指定，则会在分析模块启动前提示输入力场文件路径。

-x 此选项用于快速转换轨迹格式时使用，开启此选项则直接执行功能而不显示交互菜单。 支持输出Gromacs TRR，XTC，GRO，Amber NetCDF格式的轨迹文件。

例如

**analysis –f water.arc –x water.xtc**

将tinker的arc轨迹转换到gromacs的xtc格式轨迹

--script

--script-file

此选项使用领域特定语言（DSL）对程序进行控制，--script直接从命令行参数中读取脚本内容，而--script-file则从参数指定的文件中读取并执行脚本。

使用交互菜单和DSL执行的功能使用相同的核心计算代码，只是操作方式不同而已。

## 交互式功能

主菜单：

Main Menu

(0) Trajectory Analysis

(1) Print Topology

暂时只分为两类，0选项是传统的对轨迹分析的功能，1选项是查看拓扑文件的内容（这对于tpr这类二进制拓扑文件最为有用）

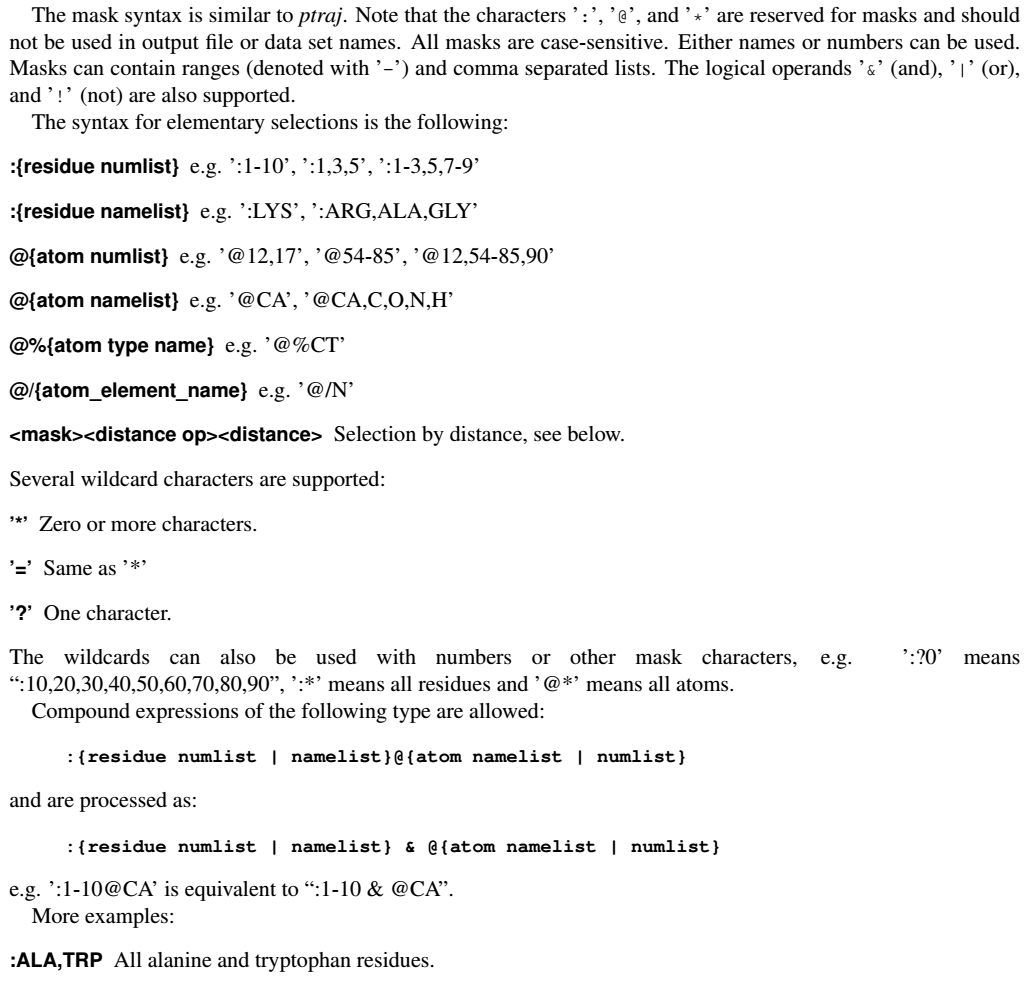
* + 1. 选项1: Print Topology

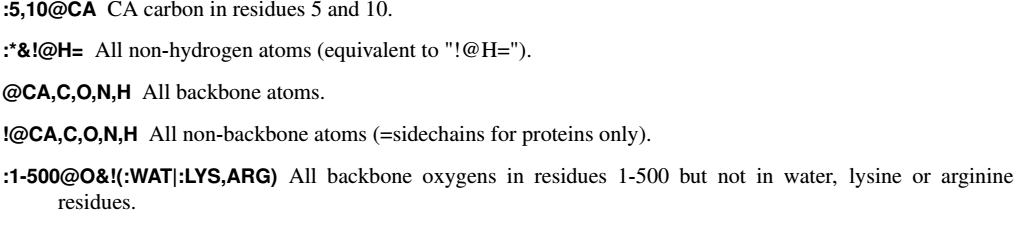
选择此选项将直接进入命令行模式，目前提供的可执行的命令有以下几类：

|  |  |
| --- | --- |
| 1. com of ambermask | 查看选中原子及其的质心 |
| 2. geom of ambermask | 查看选中原子及其的几何中心 |
| 3. ambermask | 查看选中原子 |
| 4. help | 查看帮助性质，暂时只有一张表 |
| 5. quit | 退出程序 |

为了自由地在分析中选择特定的原子和残基，基于amber的原子选择语法ambermask并对其进行了扩展。在整个程序需要选择原子的地方统一使用相同的语言。

以下语法定义截图自amber手册





唯一的区别在于&符号不可省略，比如 **:1-10@CA**在Amber中可以接受，但是在这里需要写成完整的形式 **:1-10&@CA**，此项考虑是为了简化语法分析，尽量避免上下文相关的语法。除了以上的语法，增加了 **#** 的用途, 在numlist后加入步长的需要, #后面跟的是步长（步长不能为零，没有#相当于步长为1）

例如 **@1-10#2** 相当于**@1,3,5,7,9**

另外，type列表可以是数字，即增加了语法@%{atom type numlist} e.g. @%1-10

为了支持Tinker软件中原子类型用数字代表的情况。

在Print Topology 命令行中输入ambermask后，将会输出选择的原子的列表（此功能适合在不确定ambermask表达式究竟选择了那些原子时使用，以避免在分析中选择错误的分析目标）

例如：

**> @Np**

**#Atom Name #Res Name #Mol Type Charge Mass X(Ang) Y(Ang) Z(Ang)**

**1 Np 1 Np 1 Np 4.0000 - 3.362 20.964 14.363**

程序将输出相应的所有名字为Np的原子信息列表

**> com of :1**

**#Atom Name #Res Name #Mol Type Charge Mass X(Ang) Y(Ang) Z(Ang)**

**1 O1 1 C27 1 o -0.8105 16.0000 26.558 7.509 20.517**

**2 P1 1 C27 1 p5 1.0294 30.9700 25.712 8.313 19.636**

**3 C1 1 C27 1 c3 -0.0512 12.0100 26.580 8.788 18.029**

**4 H1 1 C27 1 hc -0.0567 1.0080 26.038 8.387 17.234**

**…**

**51 H32 1 C27 1 hc 0.0378 1.0080 23.960 3.466 22.020**

**52 H33 1 C27 1 hc 0.0378 1.0080 24.702 4.853 21.418**

**53 H34 1 C27 1 hc 0.0378 1.0080 24.022 3.758 20.225**

**Mass Center X(Ang) Y(Ang) Z(Ang)**

**25.799 7.761 19.288**

程序将输出相应的所有属于第一个残基的原子信息列表，及其质量中心（暂时没有应用PBC）

**> geom of :1**

**#Atom Name #Res Name #Mol Type Charge Mass X(Ang) Y(Ang) Z(Ang)**

**1 O1 1 C27 1 o -0.8105 16.0000 26.558 7.509 20.517**

**2 P1 1 C27 1 p5 1.0294 30.9700 25.712 8.313 19.636**

**3 C1 1 C27 1 c3 -0.0512 12.0100 26.580 8.788 18.029**

**4 H1 1 C27 1 hc -0.0567 1.0080 26.038 8.387 17.234**

**…**

**51 H32 1 C27 1 hc 0.0378 1.0080 23.960 3.466 22.020**

**52 H33 1 C27 1 hc 0.0378 1.0080 24.702 4.853 21.418**

**53 H34 1 C27 1 hc 0.0378 1.0080 24.022 3.758 20.225**

**Geom Center X(Ang) Y(Ang) Z(Ang)**

**25.791 7.622 19.139**

程序将输出相应的所有属于第一个残基的原子信息列表，及其几何中心（暂时没有应用PBC）

* + 1. 选项0: 轨迹分析功能（Trajectory Analysis）[此菜单为使用Boost.MPL生成]

目前存在31个功能，但是有些功能比较基础，还有些仍在完善中。

选择相应的功能并且输入完所需的参数后将回到此菜单，可以选择在一次轨迹读取过程中进行多个分析，最后选择0输入起始和所需要读取的frame数，以及并行所需要和核数，输入后开始分析过层，实时显示当前读取的帧数。

(0) Start

(1) Gromacs XTC & TRR & GRO & NetCDF Output

(2) Distance

(3) Coordinate Number per Frame

(4) Radical Distribution Function

(5) ResidenceTime

(6) Green-Kubo

(7) Hydrogen Bond

(8) RMSD Calculator

(9) RMSF Calculator

(10) Cluster Analysis(linkage)

(11) NMRRange Analysis

(12) Diffusion Coefficient by Einstein equation

(13) Diffusion Cutoff Coefficient by Einstein equation

(14) Water exchange analysis

(15) Rotational autocorrelation function (Cutoff)

(16) Dipole Angle

(17) Dipole Angle to Gibbs Free Energy

(18) Dipole Angle of single distance normal

(19) Dipole Angle of volume normal

(20) Shell Density function

(21) Search Interaction Residue between two groups

(22) Find Min distance between two groups

(23) Calculate demix index of two groups

(24) Dipole Angle Distribution with cutoff

(25) Dipole Angle Distribution with Axis

(26) Over Plane Angle Distribution with cutoff

(27) Plane Angle Distribution with cutoff

(28) Angle (Ow-Hw) Distribution with cutoff

(29) Rotational autocorrelation function

(30) Gibbs Free Energy of Distance Angle for AnO2

(31) Dipole Angle 3 Dimension Distribution with Axis

有些条目的功能是为了特殊体系的分析要求而特别做的，而有些只是为了实现一些小功能而已，以下主要介绍个人认为做得比较好的功能

(1) Gromacs XTC & TRR & GRO & NetCDF Output： 对当前轨迹进行PBC和格式的处理，支持输出Gromacs TRR，XTC，GRO，Amber NetCDF形式的轨迹文件，并且可以同时指定多个输出文件。

对于PBC的处理可以选择并进行处理，用与轨迹格式的转换，还有将某个原子或是分子居中，后两者将自动将对分子做结构重建。

例如：

**<- (1) Gromacs XTC & TRR & GRO & NetCDF Output ->**

**-------------------------------------------------**

**output file [empty for next]: sample.xtc**

**output file [empty for next]: sample.nc**

**output file [empty for next]:**

**PBC transform option**

**(0) Do nothing**

**(1) Make atom i as center**

**(2) Make molecule i as center**

**Choose : 2**

**Please enter one atom NO. that the molecule include: 1**

将同时输出gromacs xtc和 gromacs netcdf格式的轨迹，同时将第一个分子的几何中心居中。

(3) Coordinate Number per Frame ; 输出配位数随时间的变化

需要使用ambermask选择被配位的原子，配体上的原子，以及指定cutoff

例如

**-------------------------------------**

**<- (3) Coordinate Number per Frame ->**

**-------------------------------------**

**Enter mask for atom1 : @U**

**Parsed Abstract Syntax Tree :**

**names : U**

**Enter mask for atom2 : @1-1536&@O**

**Parsed Abstract Syntax Tree :**

**&**

**names : 1-1536**

**names : O**

**Please enter distance cutoff:3.0**

将选择原子 **@U**和**@1-1536&@O**，以及设置了Cutoff = 3.0Å . 为了确保用户输入ambermask语句确实符合所要表达的意思，程序将会把语法分析后的中间语言（IR）形式抽象语法数（AST）以一种带缩进的层次化形式打印出来。

例如：

**@1-1536&@O**

**Parsed Abstract Syntax Tree :**

**&**

**names : 1-1536**

**names : O**

该语句所表达的意思是选择编号处于1-1536 **AND** 名字是O的原子。

(4) Radical Distribution Function ： RDF的计算，输出包括RDF及其积分

(5) ResidenceTime [**parallel**]： 驻留时间的计算， 计算过程使用了TBB并行{标有[**parallel**]的功能采取并行实现，下同}

此功能主要用户计算金属离子溶剂化层中配体的驻留时间

基于的计算公式：

The mean residence time τ of water molecules in ion first hydration shell was computed as below:87



where *N*0was the total number of water molecules in the first hydration shell of the actinyl ion at time t = 0, and *P*i(t,t\*) the Heaviside function, which was 1 if the *i*th water molecule at time t was correlated with the *i*th water molecule at time 0, i.e. *Pi*(*t,t*\*) =1 only if the *i*th water molecule was in the first solvation shell at both time 0 and *t*, and it did not leave the first solvation shell for any continuous period longer than *t\**.

输入包括 中心原子【此处通常为金属离子】，配体上的原子【此处通常为水分子的Ow】， 配体层的cutoff, 以及公式所要求的t\*。

例如：

**-----------------------**

**<- (5) ResidenceTime ->**

**-----------------------**

**Enter mask for atom1 : @U**

**Parsed Abstract Syntax Tree :**

**names : U**

**Enter mask for atom2 : @1-1536&@O**

**Parsed Abstract Syntax Tree :**

**&**

**names : 1-1536**

**names : O**

**Please enter distance cutoff1:3**

**Please enter t\*: ( unit: frame)10**

计算U溶剂化层的水的驻留时间曲线R(t)， cutoff = 3.0 Å， t\* = 10frames

(6) Green-Kubo [**parallel**]： 使用Green-Kubo公式计算扩散系数， 计算过程使用了TBB并行

(12) Diffusion Coefficient by Einstein equation [**parallel**]：使用Einstein公式计算扩散系数， 计算过程使用了TBB并行

(7) Hydrogen Bond ： 氢键的计算，可以选择VMD形式或是GROMACS形式的计算公式来计算氢键

输入包括供体与受体的原子区域，氢键的计算模式【VMD or GROMACS】, 以及距离和角度的cutoff

例如：

**-----------------------**

**<- (7) Hydrogen Bond ->**

**-----------------------**

**Please enter group:@1-1536**

**Parsed Abstract Syntax Tree :**

**names : 1-1536**

**Which selector: [(1)Acceptor | (2)Donor | (3)Both]:3**

**HBond criteria:**

**(1)VMD version**

**(2)GMX version**

**choose:1**

**Donor-Acceptor Distance:3**

**Angle cutoff:35**

选择@1-1536（此处为水）同时作受体和供体，distance cutoff = 3Å， angle cutoff = 25 degree

(8) RMSD Calculator ：计算选中原子的RMSD， 当前将第一帧轨迹作为reference， 在structure fit和rsmd计算中使用同一组原子。

(9) RMSF Calculator ： 基于RMSD计算每个原子在轨迹中偏移量RMSF

(10) Cluster Analysis(linkage) [**parallel**]：使用single linkage 算法来进行簇分析，算法本身来自于gromacs，但是进行并行化处理，结果计算速度快于gromacs rms命令。 输入内容包括选择RMSD计算的原子和聚类所用的cutoff， 输入内容包括每一帧结构所属的cluster编号，每一个cluster里的frame数和相应的frame列表， 在每一个cluster中处于中间结构的frame编号及其与其他结构的平均距离【RMSD】

(11) NMRRange Analysis ： 主要对于多肽进行计算NOE

为了自动识别各种H原子属于H-alpha，H-beta，H-gamma之类的情况，程序需要设置环境变量ANALYSIS\_TOP\_PATH执行存放以top类型的文件，用以读取残基的拓扑定义，由于Tinker ARC文件中的信息太少，所以需要重新从分子的原子连接关系中推导出残基片段及每个原子的类型。

例如 GLU的top文件为Glu.top，内容为：

**1 N N 2 5 6**

**2 C C 1 3 8 9**

**3 C C 2 4 18**

**4 O O 3**

**5 H HN 1**

**6 X X 1**

**8 H H-alpha 2**

**9 C C 2 10 14 15**

**10 C C 9 11 16 17**

**11 C C 10 12 13**

**12 O O 11**

**13 O O 11**

**14 H H-beta 9**

**15 H H-beta 9**

**16 H H-gamma 10**

**17 H H-gamma 10**

**18 X X 3**第一列为原子编号，从1开始， 第二列是原子元素，X代表任意原子，第三列为原子类型，这里明确指出各种H的类型，后面是该原子所连接的其他原子的编号。所以文件描述的该残基的拓扑结构，只要在分子中找到相应的结构片段，即可识别为Glu。识别所用的为图论中的深度优先搜索算法，虽然存在大量回溯，但是性能够用。

程序启动后首先识别第一帧的结构， 分析其拓扑结构， 找出所需要计算的所有氢原子对，在后续的轨迹中不断计算相应的Distance，最后将打印出每一个对H原子的NOE值。

(13) Diffusion Cutoff Coefficient by Einstein equation ： 使用Einstein公式计算配位层内配体的扩散系数。

因为配体随着时间进出配位层， 所以程序要跟踪所有配体在每一帧的位置，找出在总是配位层内的连续坐标序列，每一次配体进入配位层之后将会启动新的跟踪过程。

(14) Water exchange analysis ：有些程序不支持动态原子的选择，例如使用Gromacs计算配位层内水相互作用的能量分解。所以需要寻找连续的水未交换的时间需要，此功能将会打印每次水分子进出配位层的事件所处的时间和那个水分子往哪个方向跨越了配位层，以及当前处于配位层内水分子编号。

(29) Rotational autocorrelation function [**parallel**] ： 计算转动自相关函数及其弛豫时间，可以指定向量的内容包括：1. 使用分子中3个点计算其法向量， 2. 定义两个原子连线的向量， 3. 分子的偶极矩向量。 同时Legendre Polynomial可选择P1 = x 或是P2 = (1/2)(3x^2 -1)。

(15) Rotational autocorrelation function (Cutoff) ： 类似于29号功能，但是用于计算配位层内的配体的转动自相关，增加了配位原子的选择和cutoff，其余功能与29一致。由于配位层内配体较少，所以该功能未并行化。

(16) Dipole Angle

(17) Dipole Angle to Gibbs Free Energy

(18) Dipole Angle of single distance normal

(19) Dipole Angle of volume normal

计算 M-Ligand 距离和Ligand Dipole之间夹角的分布，16号功能是直接输出格点中频率计数值，17号功能基于频率计数转换为Gibbs Free Energy的输出，18和19分别使用不同的归一化方式， 18号是对每一壳层分别归一，19号是体积元进行归一。

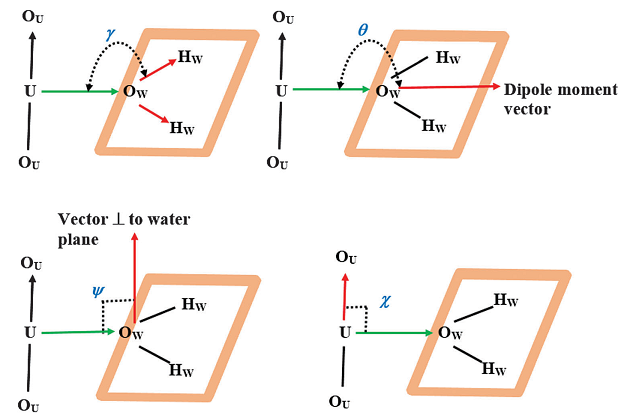
(20) Shell Density function ： 计算不同距离下配体的密度分布

(22) Find Min distance between two groups ： 计算分子间的距离，输出两个分子中原子间的最小距离。

(23) Calculate demix index of two groups ： 计算两相体系的分离指数。

类似于功能17，计算特定配位层的值，需要定义两个cutoff，只计算配位距离处于在两个cutoff之内的配体的计算值。

(25) Dipole Angle Distribution with Axis ： 计算分子的偶极矩与坐标轴夹角的分布



(24) Dipole Angle Distribution with cutoff ：计算配位区域内角度*θ*的分布密

度（上图）

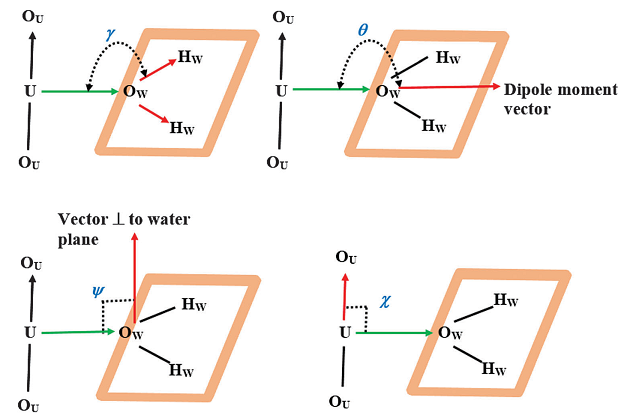
(26) Over Plane Angle Distribution with cutoff ：计算配位区域内角度*χ*的分布密

度（上图）

(27) Plane Angle Distribution with cutoff ： 计算配位区域内角度*Ψ*的分布密度（上图）

(28) Angle (Ow-Hw) Distribution with cutoff ： 计算配位区域内角度***γ***的分布密度（上图）

(30) Gibbs Free Energy of Distance Angle for AnO2 ： 用于计算下图中角度分布的自由能图



## 使用领域特定语言(DSL)

为了提高在处理大量批量分析任务时的操作效率，程序执行性能和操作的灵活性。仿照通用程序设计语言的风格，实现了C和Python语言的一个子集，并且为领域特殊用途加入了语法。

语言定义5种基本的数据类型，分别是整型(int), 浮点(double), 布尔值(bool)，字符串(string)，以及amber mask选择器(AmberMask)

字符串是使用双引号括起来的字符序列， AmberMask是使用方括号([])围起来的前述的amber mask syntax表达式。

标识符是使用字母下划线跟零到若干个字母数字下划线组成的名字。暂时在程序中用于标识变量。变量无需声明类型，与Python一致，将在赋值时自动绑定到相应的值上，而且变量具有常量性（即修改一个变量只是让其执行另一块内存单元，原地址内容不变）。变量从定义开始生效，直至程序结束。

语言暂时不能定义函数，所有函数均由内置提供，并且函数名与变量名处于不同的名字空间，因此变量名和函数名可以重名。

* + 1. 保留的关键字

以下的关键字被用做程序控制流，不能用做标识符

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **for** | **if** | **else** | **do** | **while** | **until** |

* + 1. 注释

脚本程序中支持C++风格以及Python风格的注释, 行注释从注释符开始直至行尾结束，块注释可以跨越多行但不能嵌套，（注释不能在AmberMask表达式内部使用）:

**// Line Comment**

**# Line Comment**

**/ \* Block Comment \*/**

* + 1. 支持的操作符优先级

结合性 操作符 描述

L a() Function call

---------------------------------------------------------------------------------

L a++ a-- Suffix/postfix increament and decrement

---------------------------------------------------------------------------------

R ++a --a Prefix increament and decrement

R +a -a Unray plus and minus

R ! Logical NOT

---------------------------------------------------------------------------------

L a\*b a/b a%b Multiplication, division, and remainder

---------------------------------------------------------------------------------

L a+b a-b Addition and subtraction

---------------------------------------------------------------------------------

L < <= Less, less equal relational operaters

L > >= Great, great equal relational operaters

---------------------------------------------------------------------------------

L == != Equal, not equal releational operaters

---------------------------------------------------------------------------------

L & Bitwise AND

---------------------------------------------------------------------------------

L | Bitwise OR(inclusive or)

---------------------------------------------------------------------------------

L && Logical AND

---------------------------------------------------------------------------------

L || Logical OR

---------------------------------------------------------------------------------

R = Direct assignment

R += -= Compound assignment by sum and difference

R \*= /= %= Compound assignment by product, quotient, and remainder

R &= |= Compound assignment by bitwise AND and OR

除了用于定义数据结构和面向对象所特有的操作，基本上实现了大多数的操作功能。除了以下的例外：

Bitwise Operator （& |）只能作用在AmberMask上（因为在这里实现真正的位操作没有应用）。 （& | !） 作用在AmberMask上进行如前所述的mask syntax定义的运算。

字符串的+，代表字符串的连接。 当字符串与int相加时，将后者转换位字符串再进行相加。

函数调用使用了类Pyhon的语法： function( arg1, arg2, arg3\_name = arg3, arg4\_name = arg4)

参数列表中先是位置参数， 后跟若干个named argument，后者可以乱序，但是必须在位置参数后。 若是函数存在默认参数值，则named argument可以省略，自动赋默认值，但是位置参数不可省略。 相同名字的named argument不能重复出现。

系统内置简单的通用函数,其用法与C stardard library一致

|  |  |
| --- | --- |
| 函数 | 描述 |
| int() | 强制类型转换到int |
| double() | 强制类型转换到double |
| bool() | 强制类型转换到bool |
| sqrt() | 开平方 |
| abs() | 取绝对值 |
| log() | 取对数 |
| exp() | 自然指数 |
| pow() | 幂 |
| print() | 打印值， 支持int，double和string |

* + 1. 控制语句

按照C的语法， 每条语句都是使用分号结束，使用花括号围起来的若干条语句组成语句块。

当前支持的控制流语句包括for\_loop, if\_else, while, do\_while, do\_until。其相应的扩展巴科斯范式（EBNF）如下，其语义与C语言一致：

**<for\_stmt> ::= “for” “(” <expr> “;” <condtion\_expr> “;” <expr> “)” “{” <stmts> “}”**

**<while\_stmt> ::= “while” “(” <condtion\_expr> “)” “{” <stmts> “}”**

**<if\_else\_stmt> ::= “if” “(“ <condtion\_expr> “)” “{” <stmts> “}” [ “else” “{” <stmts> “}” ]**

**<do\_while\_stmt> ::= “do” “{” <stmts> “}” “while” “(” <condtion\_expr> “)” “;”**

**<do\_until\_stmt> ::= “do” “{” <stmts> “}” “until” “(” <condtion\_expr> “)” “;”**

为了语法分析避免歧义，即使在语句块中只有一条语句，花括号也不可省略。

* + 1. 内置函数
       1. 轨迹分析函数
* **Radical Distribution Function 径向分布函数**

**rdf ( M : AmberMask, L : AmberMask, max\_dist : double = 10.0, width : double=0.01, intramol : bool = false, out : string )**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| M | AmberMask | 中心原子的AmberMask表达式（通常为金属离子） |
| L | AmberMask | 配位原子的AmberMask表达式（例如@OW） |
| max\_dist | double | 计算RDF的最大距离范围，单位为Å |
| width | double | 计算格点的宽度，单位为Å |
| intramol | bool | 是否计算分子内的的配位 |
| out | string | 分析结果输出文件名 |

* **Rotational Autocorrelation Function 转动自相关函数**

**rotacf ( vector : VectorSelector, P : int, time\_increment\_ps : double = 0.1, max\_time\_grap\_ps : double, out : string )**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| vector | VectorSelector | 计算向量的选择器，可选多种，由下面的辅助函数生成 |
| P | int | Legendre Polynomial的级别，  『 P=1， P1(x) = x 』  『 P=2， P2(x) = 』 |
| time\_increment\_ps | double | 每一帧的时间间隔，用于计算积分，单位ps |
| max\_time\_grap\_ps | double | 计算ACF的最大时间范围，因为转动较快时，ACF后半部分用不上，这样做可以加快计算速度 |
| out | string | 分析结果输出文件名 |

* **Rotational Autocorrelation Function within Solvation Shell 配位层内的转动自相关函数**

**rotacfCutoff ( M : AmberMask, L : AmberMask, vector : VectorSelector, P : int, cutoff: double, time\_increment\_ps : double=0.1, max\_time\_grap\_ps : double, out : string )**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| M | AmberMask | 中心原子的AmberMask表达式（通常为金属离子） |
| L | AmberMask | 配位原子的AmberMask表达式（例如@OW）,  每个配位分子只能有一个配位原子 |
| vector | VectorSelector | 计算向量的选择器，可选多种，由下面的辅助函数生成，用于计算被L参数选上的分子中的向量 |
| P | int | Legendre Polynomial的级别，  『 P=1， P1(x) = x 』  『 P=2， P2(x) = 』 |
| cutoff | double | 配位的截断距离，单位为Å |
| time\_increment\_ps | double | 每一帧的时间间隔，用于计算积分，单位ps |
| max\_time\_grap\_ps | double | 计算ACF的最大时间范围，因为转动较快时，ACF后半部分用不上，这样做可以加快计算速度，单位ps |
| out | string | 分析结果输出文件名 |

* **DemixIndex of Bi-component System 双组分体系的分离指数**

**demix ( component1 : AmberMask, component2 : AmberMask, grid : Grid, out : string )**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| component1 | AmberMask | 组分1所有原子的选择器 |
| component2 | AmberMask | 组分2所有原子的选择器 |
| grid | Grid | 盒子的格点的划分方式，由Grid函数生成 |
| out | string | 分析结果输出文件名 |

* **Residence Time of Ligand in Solvation Shell 驻留时间计算**

**residenceTime ( M : AmberMask, L : AmberMask, cutoff : double, time\_star : int, out : string )**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| M | AmberMask | 中心原子的选择器 |
| L | AmberMask | 配体上的配位原子选择器 |
| cutoff | double | 配位的截断距离，单位为Å |
| time\_star | int | 公式中t\*的值，单位：帧 |
| out | string | 分析结果输出文件名 |

* **Self-Diffusion Coefficient Calculation based on Einstein Equation :**

**diffuse ( mask : AmberMask, time\_increment\_ps : double = 0.1, total\_frames : int, out : string )**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| mask | AmberMask | 被计算的分子 |
| time\_increment\_ps | double | 每一帧的时间间隔，单位ps |
| total\_frames | int | 总共需要计算的帧数，为了计算高效，需事先知道所需的总内存大小，单位：帧 |
| out | string | 分析结果输出文件名 |

* **Self-Diffusion Coefficient Calculation based on Einstein Equation within Solvation Shell 配位层内配体的扩散系数:**

**diffuseCutoff (M : AmberMask, L : AmberMask, cutoff : double, time\_increment\_ps : double = 0.1, out : string )**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| M | AmberMask | 中心原子，整个体系只能有一个(通常为金属) |
| L | AmberMask | 被计算的分子上的配位原子，每个分子只能有一个，用与计算配位距离 |
| cutoff | double | 配位的截断距离，单位为Å |
| time\_increment\_ps | double | 每一帧的时间间隔，单位ps |
| out | string | 分析结果输出文件名 |

* + - 1. 辅助函数

* **向量选择器**

**VectorSelector DipoleVector ( mask : AmberMask )**

**VectorSelector TwoAtomVector ( mask1 : AmberMask, mask2 : AmberMask )**

**VectorSelector NormalVector ( mask1 : AmberMask, mask2 : AmberMask, mask3 : AmberMask )**

向量的选择器，分别是偶极矩，以双原子连线的向量，三个原子定义平面的法向量，返回**VectorSelector**。

* **格点选择器**

**Grid grid ( x : int, y : int, z : int )**

空间格点，参数为x，y，z轴方向上需要切分的格点数，返回**Grid**类型的三元组。

* **开始计算函数**

**int go ( start : int = 1, end : int = 0, step : int = 1, nthreads = 0 )**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| start | int | 开始计算的帧 |
| end | int | 结束计算的帧，0代表轨迹文件结束 |
| step | int | 读取的步长 |
| nthreads | int | 并行所用核数，0代表自动选择（通常为系统CPU核心数） |

所有分析函数的执行只是加入了一个待执行的任务，其本身需要等待go函数驱动最后的执行。成功执行此指令之后将清空待计算的队列，并且返回完成计算的任务数。对于存在多个待计算的任务，使用了任务间并行，多个只能串行执行的作业可以并行执行。并行可以嵌套，子任务也可以并行，程序将尽最大可能利用计算核心。

由于同时计算所有需要的作业需要的内存超出机器的资源，在脚本中可以使用多个go语句，每次执行将重新读取轨迹计算当前任务。

* **拓扑文件设置函数**

**readTop ( file : string )**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| file | string | 拓扑文件名 |

未在命令行参数中指定拓扑文件的情况下，可以使用此函数在脚本开头指定所需的拓扑文件，若程序需要但未指定，将打印错误信息并退出。重复使用此函数将覆盖上次拓扑文件名。

* **轨迹文件设置函数**

**trajin ( file : string )**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| file | string | 轨迹文件名 |

未在命令行参数中指轨迹文件的情况下，可以使用此函数在脚本开头指定所需的轨迹文件，若程序需要但未指定，将打印错误信息并退出。重复使用此函数将按照函数执行的顺序指定多个轨迹文件。

* **力场文件设置函数**

**readFF ( file : string )**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| file | string | 力场文件名 |

未在命令行参数中指力场文件的情况下，可以使用此函数在脚本开头指定所需的力场文件，若程序需要力场文件但未指定，将打印错误信息并退出。重复使用此函数将覆盖上次力场文件名。当前只支持Tinker 的prm格式的力场文件，配合tinker xyz的拓扑文件使用。

# 实现细节

## 语法分析器与解释器

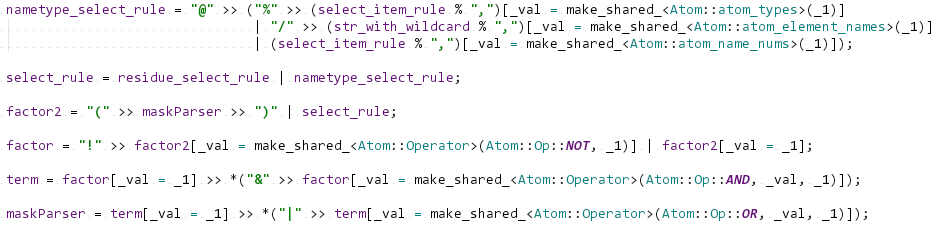
Ambermask的语法分析器（Parser）和生成器（Generator）分别使用了Boost.Spirit.Qi 和Boost.Spirit.Karma进行实现。前者使用了递归下降LL(∞)算法，使用了扩展巴科斯范式（EBNF）进行语法描述，但是为了避免回溯，做到LL(1)的效果，因此在语法分析上将上下文相关的内容将在语义分析阶段进行，即先生成带有歧义的与语法分析结果再进行转换。分析器的实现文件在src/grammar.hpp。

生成器的功能为将抽象语法树转化为源语句本身，由于在语法分析的过程损失了一些对语义无关的信息，所以生成器生成的内容只是与输入的内容在语义上是一致的，但是表达式字符串并不一定相等。分析器的实现文件在src/ GeneratorGrammar.hpp。

在src/Interpreter.hpp中定义了2.3节中实现的小型DSL的语法分析器（包含了AmberMask的语法分析器）。src/Interpreter.hpp后半步分和src/Interpreter.cpp中为解释器核心的实现代码，语法分析和解释器核心共计一千行代码。

以下是整个语法分析器使用Boost.Spirit.Qi表达式模板实现的语法制导翻译方案（SDT），包含大部分语义分析代码：

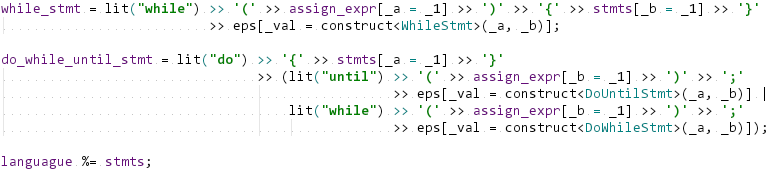












由于AmberMask语法简单而且固定，所以使用了boost.variant, 而整个分析器暂时使用了boost.any作为语法节点的容器，所以有一些重复的操作符的定义。

在语法分析方面的问题在于无法在脚本写错的时候报告准确的错误位置和原因，或是只能在解释运行的时候才能发现程序的问题。一方面在于动态类型造成的难以对语言进行静态的语义分析，另一方面在于龙书中缺乏错误处理理论和算法的描述。

## 分析功能的模块化



所有的轨迹分析的功能都聚集在相应的类中，其中轨迹分析的接口为BasicAnalysis纯虚类，所有轨迹分析功能都需要直接或间接继承这个类，实现其虚函数。完成每个虚函数所要求的功能后在taskMenu.cpp中注册一个类，然后程序框架将会读取轨迹，并且在适当的时候调用分析类的相应虚函数，完成轨迹分析。这样做的话，实现新的分析功能，只需要考虑如何实现分析算法，而不用考虑输入输出的内容，不用关心拓扑文件和轨迹文件的格式问题。

# 程序需要一步完善的地方

## 增加测试覆盖

由于程序一开始并没有对可测试有任何考虑，所以现在存在许多测试并未覆盖的地方，但是为了提高功能的可靠性，需要分离一些模块的功能，需要编写功能代码的测试用例规模去覆盖功能代码区域，提高测试覆盖率。

## 使用领域特定语言(DSL)

由于前期功能比较简单，所以采用了Multiwfn类似的使用数字来选择功能，但是随着需要分析的数据越来越多，交互式界面效率过低。

当前实现的脚本控制语言主要是通用程序语言的一个子集，其缺乏领域特定的应用方式，需要提供一种提供类似SQL的声明式的第四代语言。